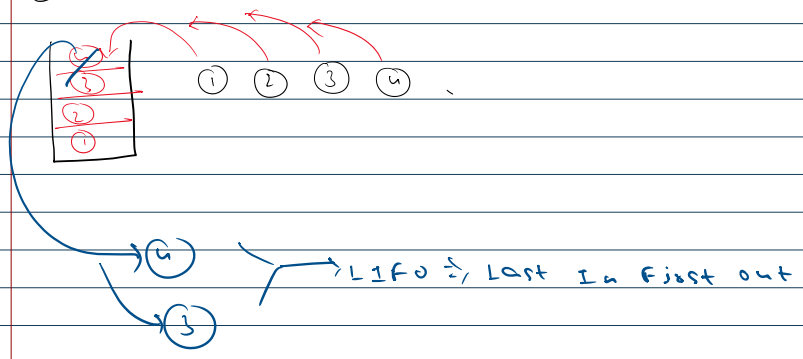
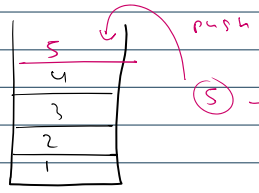
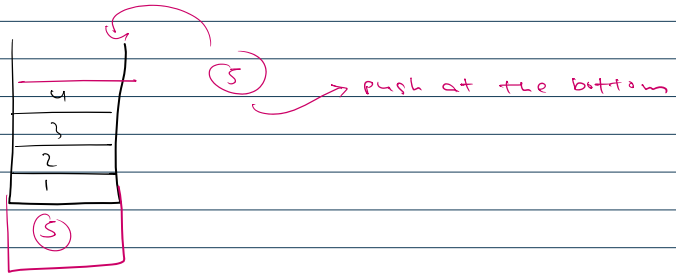
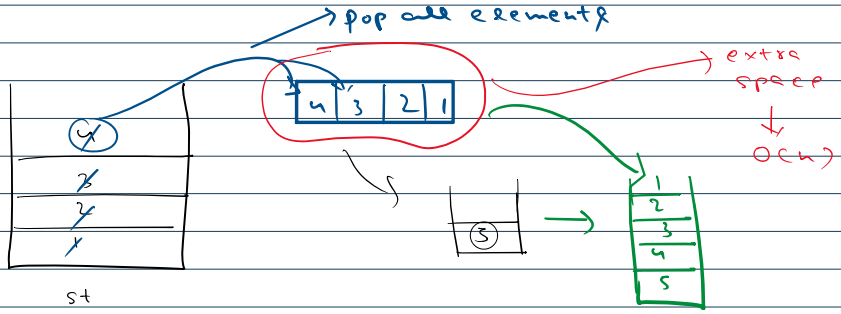
Stack



Add an element at the bottom of the stack



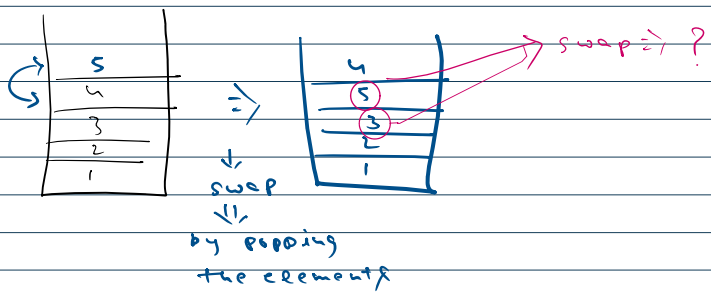




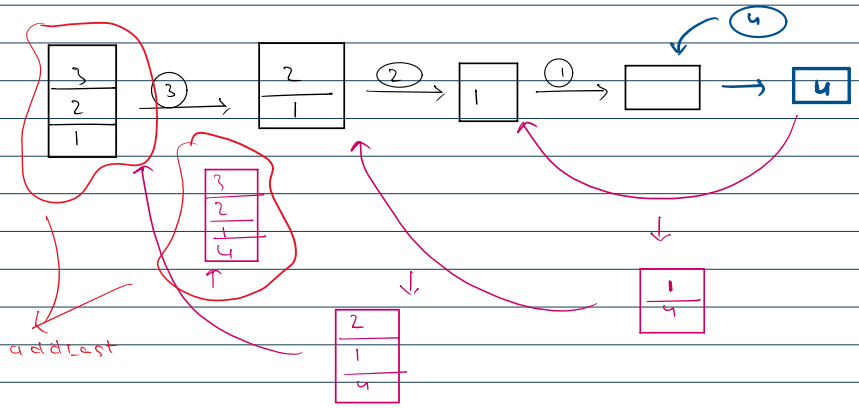

```
public class Main {
    public static void main(String[] args) {
        Stack<Integer> st = new Stack<>();
        st.push(1);
        st.push(2);
        st.push(3);
        st.push(4);
        st.push(5);
        System.out.println(st);

        addLast(st, 6);
        System.out.println(st);
    }

    public static void addLast(Stack<Integer> st, int item){
        if(st.isEmpty()){
            st.push(item);
            return;
        }
        int alpha = st.pop();
        addLast(st, item);
        st.push(alpha);
    }
}
```

1) Extra space → ✓

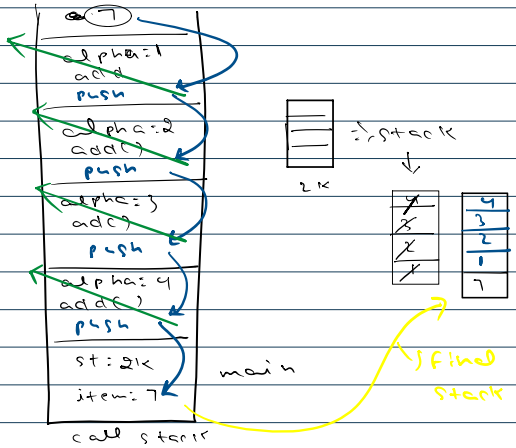
2) Rotate the element by swapping => ✗ = Indexing is not defined



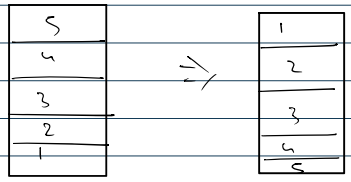
Recursive code

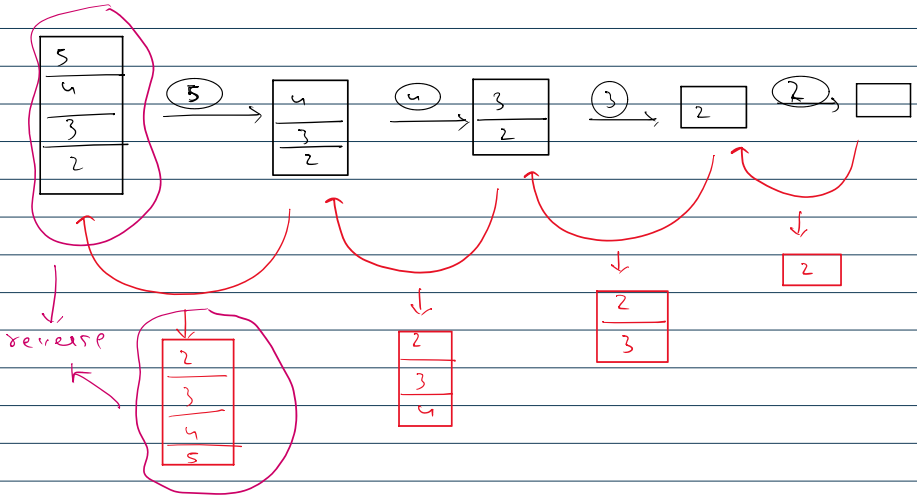



```java
public static void addLast(Stack<Integer> st, int item){
    if(st.isEmpty()){
        st.push(item);
        return;
    }
    int alpha = st.pop();
    addLast(st, item);
    st.push(alpha);
}
```



Reverse the stack






```java
public class Main {
    public static void main(String[] args) {
        Stack<Integer> st = new Stack<>();
        st.push(1);
        st.push(2);
        st.push(3);
```

```
        st.push(4);
        st.push(5);
        System.out.println(st);

        addLast(st, 6);
        System.out.println(st);

        reverse(st);
        System.out.println(st);
    }

    public static void addLast(Stack<Integer> st, int item){
        if(st.isEmpty()){
            st.push(item);
            return;
        }
        int alpha = st.pop();
        addLast(st, item);
        st.push(alpha);
    }

    public static void reverse(Stack<Integer> st){
        if(st.isEmpty()){
            return;
        }
        int beta = st.pop();
        reverse(st);
        addLast(st, beta);
    }
}
```

2375. Construct Smallest Number From DI String

Solved

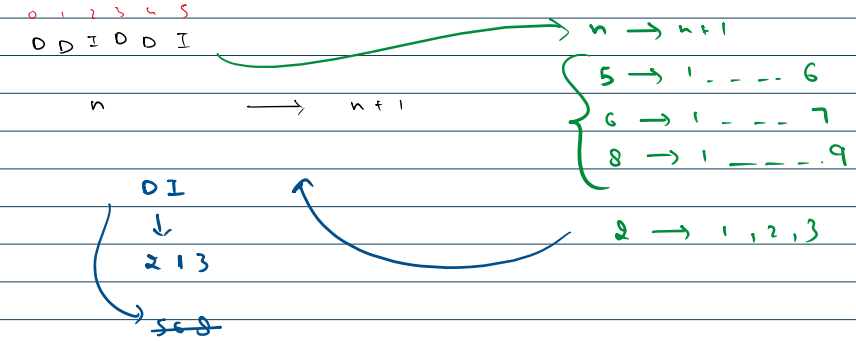
Medium · Topics · Companies · Hint

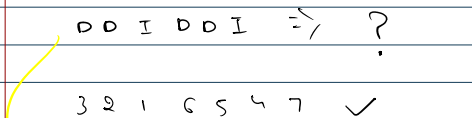
You are given a **0-indexed** string `pattern` of length `n` consisting of the characters `'I'` meaning **increasing** and `'D'` meaning **decreasing**.

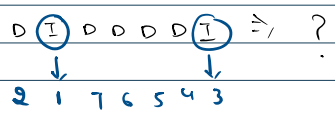
A **0-indexed** string `num` of length `n + 1` is created using the following conditions:

- `num` consists of the digits `'1'` to `'9'`, where each digit is used **at most** once.
- If `pattern[i] == 'I'`, then `num[i] < num[i + 1]`.
- If `pattern[i] == 'D'`, then `num[i] > num[i + 1]`.

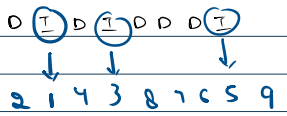
Return *the lexicographically* ***smallest*** *possible string* `num` *that meets the conditions.*

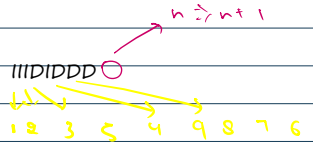




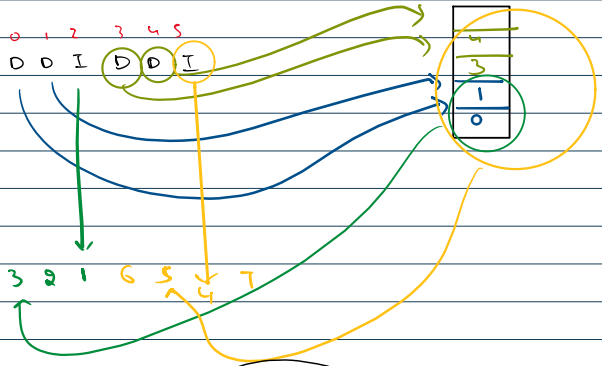


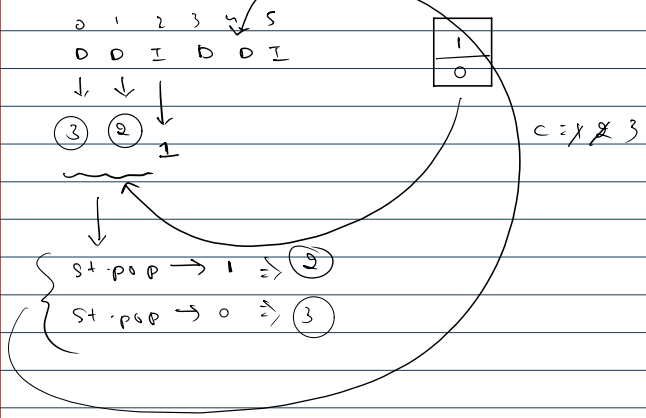

```
public class Main {
    public static void main(String[] args) {
        String str = "IIIDIDDD";
        System.out.println(smallestNumber(str));
    }

    public static String smallestNumber(String str){
        Stack<Integer> st = new Stack<>();
        int[] ans = new int[str.length()+1];
        int c = 1;
        for(int i=0; i<=str.length(); i++){
            if(i==str.length() || str.charAt(i) == 'I'){
                ans[i] = c;
                c++;

                while(!st.isEmpty()){
                    int alpha = st.pop();
                    ans[alpha] = c;
                    c++;
                }
            }
            else{
                st.push(i);
            }
        }

        String s = "";
        for(int i=0; i<ans.length; i++){
            s = s + ans[i];
        }
        return s;
    }
```



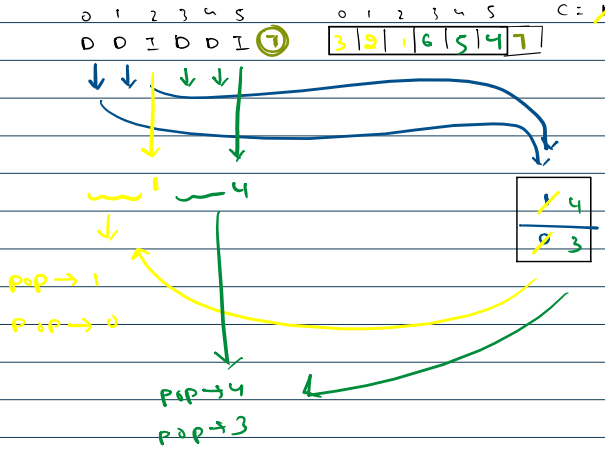


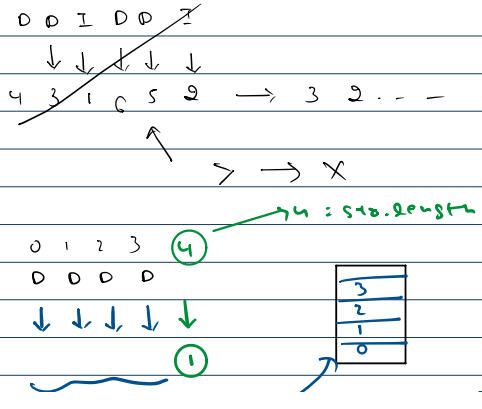
```
class Solution {
    public String smallestNumber(String str) {
        Stack<Integer> st = new Stack<>();
        int[] ans = new int[str.length()+1];
        int c = 1;
        for(int i=0; i<=str.length(); i++){
            if(i==str.length() || str.charAt(i) == 'I'){
                ans[i] = c;
                c++;

                while(!st.isEmpty()){
                    int alpha = st.pop();
                    ans[alpha] = c;
                    c++;
                }
            }
            else{
                st.push(i);
            }
        }

        String s = "";
        for(int i=0; i<ans.length; i++){
            s += ans[i];
        }
        return s;
    }
}
```

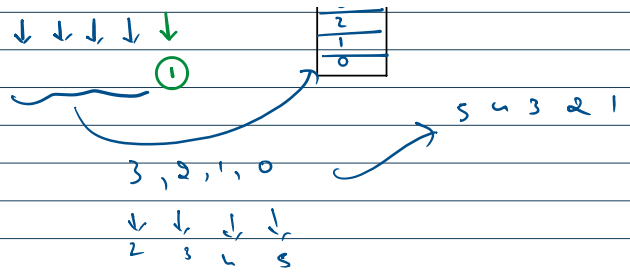




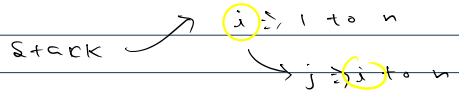
```
public static String smallestNumber(String str){
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[str.length()+1];
    int c = 1;
    for(int i=0; i<=str.length(); i++){
        if(i==str.length() || str.charAt(i) == 'I'){
            ans[i] = c;
            c++;

            while(!st.isEmpty()){
                int alpha = st.pop();
                ans[alpha] = c;
                c++;
            }
        }
        else{
            st.push(i);
        }
    }
```

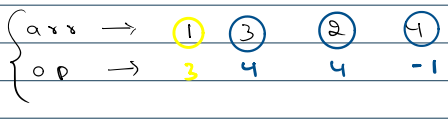


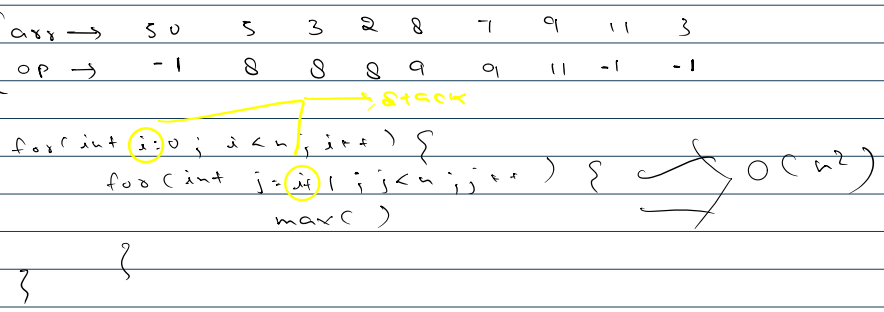




Next largest element to right

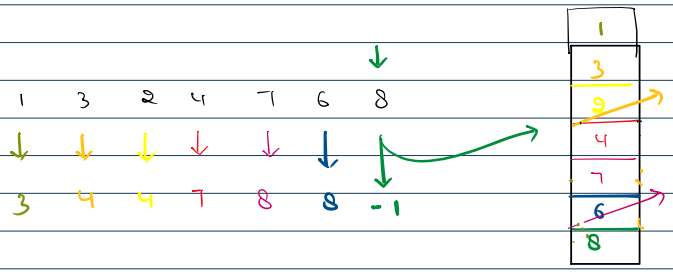


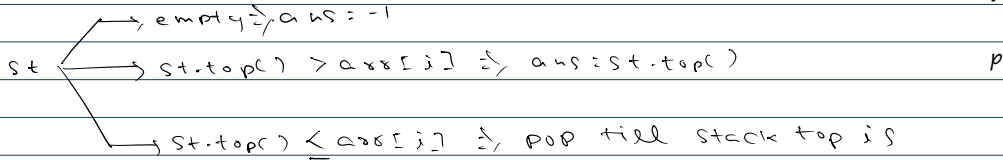


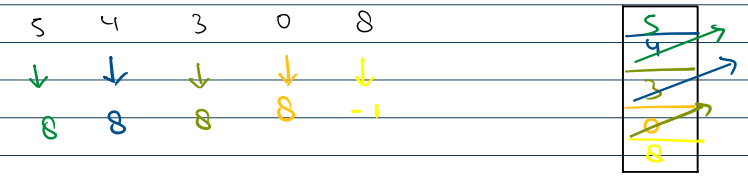


Algorithm

st.top :> largest element







```
public class Main {
    public static void main(String[] args) {
        int[] arr = {4, 5, 2, 10, 12, 7};
        int[] ans = nger(arr);
        display(arr);
        display(ans);
    }

    public static int[] nger(int[] arr){
        Stack<Integer> st = new Stack<>();
        int[] ans = new int[arr.length];

        for(int i=arr.length-1; i>=0; i--){
            if(st.isEmpty()){
                ans[i] = -1;
            }
            else if(st.peek() > arr[i]){
                ans[i] = st.peek();
            }
            else{
                while(!st.isEmpty() && st.peek() <= arr[i]){
                    st.pop();
                }
                if(st.isEmpty()){
                    ans[i] = -1;
                }
                else{
                    ans[i] = st.peek();
                }
            }
            st.push(arr[i]);
        }
        return ans;
    }

    public static void display(int[] a){
        for(int i=0; i<a.length; i++){
            System.out.print(a[i] + " ");
        }
        System.out.println();
    }

}
```

public class Main {

```
public static void main(String[] args) {
    int[] arr = {4, 5, 2, 10, 12, 7};
    int[] ans1 = nger(arr);

    display(arr);
    display(ans1);

    int[] ans2 = ngel(arr);
    display(ans2);

    int[] ans3 = nler(arr);
    display(ans3);

    int[] ans4 = nlel(arr);
    display(ans4);
}

public static int[] nger(int[] arr){
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[arr.length];

    for(int i=arr.length-1; i>=0; i--){
        if(st.isEmpty()){
           ans[i] = -1;
        }
        else if(st.peek() > arr[i]){
            ans[i] = st.peek();
        }
        else{
            while(!st.isEmpty() && st.peek() <= arr[i]){
                st.pop();
            }
            if(st.isEmpty()){
                ans[i] = -1;
            }
            else{
                ans[i] = st.peek();
            }
        }
        st.push(arr[i]);
    }
    return ans;
}

public static int[] ngel(int[] arr){
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[arr.length];

    for(int i=0; i<arr.length; i++){
        if(st.isEmpty()){
           ans[i] = -1;
        }
        else if(st.peek() > arr[i]){
            ans[i] = st.peek();
        }
        else{
            while(!st.isEmpty() && st.peek() <= arr[i]){
                st.pop();
            }
            if(st.isEmpty()){
```

```
                ans[i] = -1;
            }
            else{
                ans[i] = st.peek();
            }
        }
        st.push(arr[i]);
    }
    return ans;
}

public static int[] nler(int[] arr){
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[arr.length];

    for(int i=arr.length-1; i>=0; i--){
        if(st.isEmpty()){
            ans[i] = -1;
        }
        else if(st.peek() < arr[i]){
            ans[i] = st.peek();
        }
        else{
            while(!st.isEmpty() && st.peek() >= arr[i]){
                st.pop();
            }
            if(st.isEmpty()){
                ans[i] = -1;
            }
            else{
                ans[i] = st.peek();
            }
        }
        st.push(arr[i]);
    }
    return ans;
}

public static int[] nlel(int[] arr){
    Stack<Integer> st = new Stack<>();
    int[] ans = new int[arr.length];

    for(int i=0; i<arr.length; i++){
        if(st.isEmpty()){
            ans[i] = -1;
        }
        else if(st.peek() < arr[i]){
            ans[i] = st.peek();
        }
        else{
            while(!st.isEmpty() && st.peek() >= arr[i]){
                st.pop();
            }
            if(st.isEmpty()){
                ans[i] = -1;
            }
            else{
                ans[i] = st.peek();
            }
        }
```

```
            st.push(arr[i]);
        }
        return ans;
    }

    public static void display(int[] a){
        for(int i=0; i<a.length; i++){
            System.out.print(a[i] + " ");
        }
        System.out.println();
    }

}
```